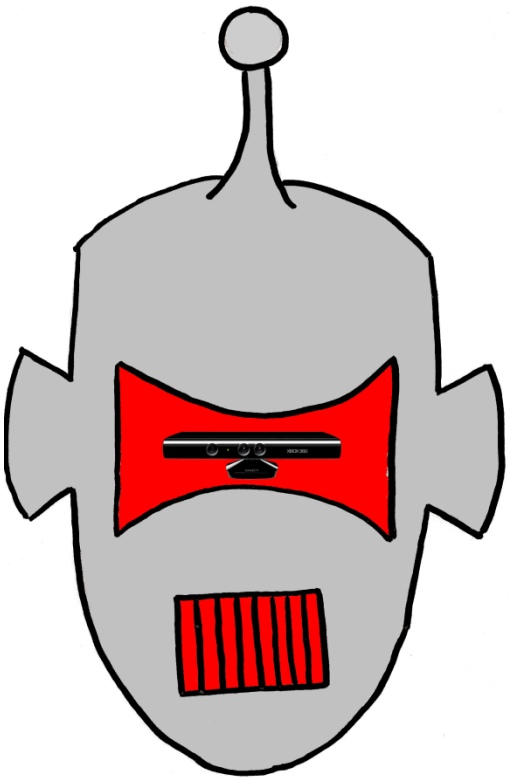


HACK ATTACK

(Kinect Game)

Technical Report

Name: Sam Duff

Student No: x08794715

Summary

This project is a requirement for my degree course titled BSc (Hons) in Software Systems.
I chose to explore game development using the Kinect Sensor. The Kinect is a motion sensing input device.

Declaration Cover Sheet for BSHSS4 Project Submission

SECTION 1 *Student to complete*

Name:
Student ID:
Supervisor:

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature:_____ Date:_____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks

and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced
- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

Contents

1 Executive Summary 9

2 Introduction 10

2.1 Background 10

2.2 Aims 11

2.3 Technologies 11

2.4 Structure 12

3 System 14

3.1 Requirements 14

3.1.1 Functional requirements 14

3.1.2 Non Functional requirements 14

3.1.3 Data requirements 14

3.1.4 User requirements 14

3.1.5 Environmental requirements 15

3.1.6 Usability requirements 15

3.2 Design and Architecture 16

3.3 Implementation 16

3.3.1 Prototype Implementation 17

3.3.2 Final Submission Implementation 17

3.3.3 Other Areas Explored 18

3.3.4 Code tried that was not finished 18

3.4 Testing 18

3.5 Graphical User Interface (GUI) Layout 19

3.6 Customer testing 20

3.7 Evaluation 21

4 Conclusions 22

5 Further development or research 23

6 Bibliography 24

7 Appendix 25

7.1 Project Proposal 25

TITLE 25

Kinect Game (Title Undecided) 25

Objectives 26
Background 26
Technical Approach 26
Special resources required 27
Project Plan 28
Technical Details 29
Evaluation 29
Consultation 1 29
Consultation 2 29
Proposed Supervisor 29
7.2 Requirement Specification 31
Title Requirements Specification (RS) 31
Introduction 34
Purpose 34
Project Scope 34
Definitions, Acronyms, and Abbreviations 34
User requirements definition 34
System architecture 34
Requirements specification 35
Physical environment requirements 35
Requirement 1 <Play Space> 35
Description & Priority 35
Requirement 2 <Lighting> 35
Description & Priority 35
Interface requirements 35
Functional requirements 35
Requirement 1 <Start Game> 35
Requirement 2 <View High Score> 36
Requirement 3 <Change Avatar> 36
Documentation requirements 36
Requirement 1 <User Manual> 36
Description & Priority 36

Non-Functional Requirements 36
Performance/Response time requirement 36
System evolution 36
Appendices 37
Use case diagram 37
Use case 1 37
Use case 2 38
Use case 3 40
Figure A 42
7.3 Monthly log book 43
7.4 Other material Used 46
7.5 Prototype Code 47
7.5.1 MainWindow.xaml.cs 47
7.5.2 MainWindow.xaml 50
7.5.3 GUI Layout for Prototype 51
7.6 Attempted Code 52
7.6.1 Input.cs 52
7.6.2 Menu.cs 53
7.7 Drawn Images that where scanned and edited with gimp 57

1 Executive Summary

Hack Attack is a 2D Kinect game that was built using XNA. The game is about a robot trying to kill off all of the virus's within a computer mainframe. The player interacts with the game by using their own body movements and the on screen avatar copies them. He must use his fists to punch the virus away.

The game was developed by using the XNA framework and the Kinect SDK. XNA allows for games to be built in C#. Game development can take place for windows, windows mobile and Xbox 360. The gaming market is constantly increasing. Emerging tools like Kinect and XNA can be used to build interactive games.

2 Introduction

In this report I will talk about Hack Attack a game I built that uses Microsoft's Kinect Sensor. The game tracks the player's body movements by pinpointing 20 joints on the player's body. An onscreen avatar copies all body movements made. Therefore there is no need for a controller as your body acts as a controller.

In this document I will go through the background of why I chose this topic and the aims the project should complete. I will talk about the technologies that were used and the system requirements that the project should comply to. Other topics I will cover will include the design and architecture of the system. How the system was implemented and the testing that was carried out which includes customer testing. An evaluation of the system and finally my conclusions to the project.

I will also include a bibliography to reference were my research was taken from. There will also be an appendix which will include learner journals, my original project proposal, requirements speciation and any other material which may have been used.

2.1 Background

There are a number of reasons why I decided to develop the above mentioned game. The first reason was my keen interest in gaming. Game development is an area I would like pursue as my future career. I feel this is an area I will enjoy to work in but also even in the current economic downturn there is a rapidly growing market in the gaming area. If we look at the market share figures from the US for gaming we can see that in 2004 the market share was worth USD$10.3 billion were as in 2011 the market share was worth USD$65 billion.

I also attended the GDC (Game Development Conference) Ireland event in the Gibsion Hotel on the February 15 2012. I knew of some game development companies in Ireland but had the opportunity see many more and to hear about their current projects and speak to some of them in person. Such companies include RedWind, Havock and Popcap.

Another reason for choosing the specific area of a Kinect game was to push myself and learn about a new up and coming technology. The Kinect was only released November 4, 2010. The beta version of the SDK was only released on the June 16, 2011 and the official version was only released Feb 1, 2012. Kinect actually hold a Guinness world record for being the "Fastest selling consumer electronics device" They sold a massive 8 million units in the first 60 days of release.

My console of choice is the Xbox and I have had many hours of fun playing with the Kinect games available. I have witnessed first-hand the easy at which both young and old have when using Kinect and also how much fun is involved. When Kinect first came on the market I could see the potential it had and now a year down the line it has started taking full advantage of the capability that the technology has.

Other Technologies where considered like the Wii or Playstation Move which both use motion technologies. But these technologies did not have as much functionality that could be used whereas with the Kinect you track many different elements with no need for a hand held controller. These can include skeletal tracking, depth precision, voice control and a standard camera. Due to the fact that the Kinect is Microsoft technology there is a good possibility in the future that it will have a great impact on PC's as well.

2.2 Aims

The aim of the project was to build a 2D interactive game. This game would be more aimed towards children but can be played by all age groups. The reasoning behind this is that in my personal experiences older generations are happy to sit holding a controller in their hand whereas younger kids appreciate being interactive with the game and would be have more fun jumping around while a game. The aim would be to have an on screen avatar that would copy all movements made by the player. The player would be able to interact with objects that appear on the screen without the need of a controller. The aim is that it will be an easy game to use but still fun to play.

2.3 Technologies

The Kinect motion sensor was the only piece of technology that did not change during the course of the project. Due to the fact that this is such new technology and a lot of the software was still only been released during the time span I had to develop this project many different SDK's and software were used to develop this game.

Technologies used in the project:

- Kinect Motion Sensor
- Open NI
- Sensor Kinect
- NITE

These are the early technologies used in the project and are all open source technologies which can be used with any form of OS; Windows, Mac, etc.

- Kinect SDK Beta Version
- Visual Studio 2010 C#
- .NET Framework
- DirectX
- Microsoft Platform

These were used for the prototype of the project it was created as a .wpf (windows presentation foundation) project.

- Kinect SDK Official version
- Microsoft C# 2010
- C# Language
- Microsoft XNA 4.0 framework
- Microsoft Visual Studio 2010 Express for Windows Phone
- Gimp

In the current version of my project due to the fact that the official SDK release date only happening in February these technologies were used for the final submission.

I feel I was very limited in the software that was available to me when I first started this project. There were very few resources available during the start of development as this was such a new technology. There were no books released on Kinect development until January 2012. Open NI was the first choice for development that I looked into as this was developed by the hacker community. But due to this fact Microsoft released a Beta version of their SDK which was used in the prototype. The reason for choosing this was it was less likely to damage the hardware and because it was official better resources available in future.

When the official SDK was released it allowed for the development of Kinect project using XNA. This meant that using XNA any games developed could be deployed to the Xbox 360 Console which is what the Kinect was originally developed for. There is an XNA indie games market place but this is only available in the USA not in Ireland. XNA is a well-developed framework that allows for easy development of games that can be used on multiple windows platforms.

2.4 Structure

The following sections in this document are as follows:

System Requirements:
Here I will look at Functional Requirements, Non-functional Requirements, Data Requirements, User Requirements, Usability Requirements and Environmental Requirements of the system.

System Design and Architecture:
This section describes the system design pattern and the overall system architecture.

System Implementation:
Here I will describe the different stages of my development, finial submission, prototype submission, non- working code and other tutorials used.

System Testing:
This section details the methods used to test the system

Graphical User Interface (GUI) Layout:
This is where I describe the system GUI and provides diagrams of such.

Customer Testing:
I will show the methods used for customer testing and results acquired from such.

System Evaluation:
Here I will evaluation of the system and how this evaluation was conducted.

Conclusion:
Here I will detail the conclusion of the project such as the advantages and disadvantages, opportunities and limitation of the project.

Further Development and Research:
This section will discuss further development and research possibilities.

Bibliography and Appendix:
The document concludes with a bibliography and Appendix.

3 System

3.1 Requirements

The following section show all form of requirements of the system including functional, Nonfunctional, Data, User and Environmental requirements.

3.1.1 Functional requirements

The game should effectively load all required resources such as sprites.
Images for skeletal tracking should not appear on the screen until and human form is detected.
Error messages should occur if there is a problem with the Kinect sensor.
The game should correctly identify collision detection to remove sprites when interacted with.

3.1.2 Non Functional requirements

The following specifies the nonfunctional requirements for the system.

Response time is an important factor as the game will involve the user to react to moving objects on screen. If there is a delay between the user movements and game movements the user will probably be frustrated and will not enjoy the game.

Response Time: Player movements should happen instantaneously on screen. There should be split second delay between player's movement and movement on screen.

System performance: Error Detection messages to inform if there is a problem e.g., Kinect not detected.

Game Architecture: Should be easily manipulated for future development.

3.1.3 Data requirements

There are no data requirements needed as no information is stored.

3.1.4 User requirements

For an Xbox 360 build of the project

- Kinect, Xbox 360 console unit
- Subscription to Xbox Live and the XNA creator club
- A PC from which to transfer the build.

**There is no Xbox build in this submission.*

For Windows build of the project

- A PC with Windows 7 Operating System with a minimum of 2GB of RAM
- Microsoft Visual Studio 2010 Express for Windows Mobile installed
- XNA 4.0 framework installed.

3.1.5 Environmental requirements

A minimum of 6 feet between the player and the Kinect sensor is required.

Adequate lighting is required for the senor to pick you up correctly.

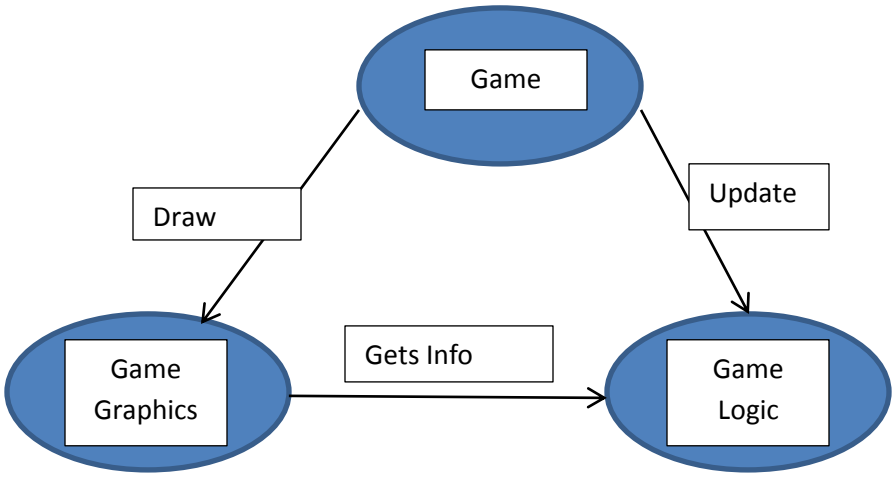
As movement is required during game play a clear play space is required as you do not want to be fall over or bang in any objects that may be in your way. Also ensure that others remain out of harm's way.

3.1.6 Usability requirements

Unfortunately this game is not disabled friendly as you need to be able to move around freely and on occasion jump to reach targets. Otherwise this game is useable for the ages 3 and up.

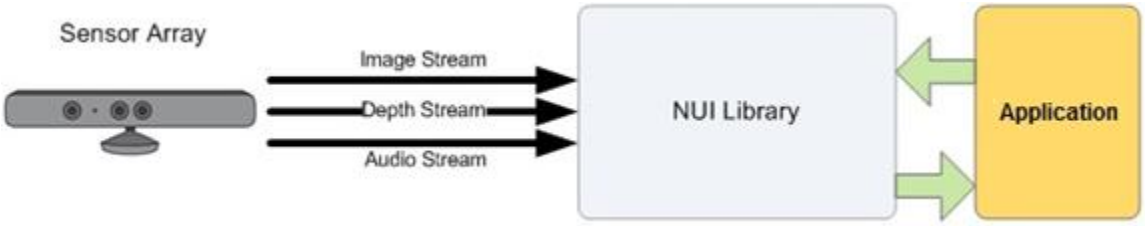
3.2 Design and Architecture

The core of my architecture is based on the classical Model View Controller pattern. Instead of the controller being the human it is the XNA Game class that controls everything.



Two main game loops take place that the game class executes. One of them updates the logic of the game and the other one draws what the player can see. For example to draw the skeleton the game needs to know where to draw the skeleton. Can it be drawn at that position. This is how the logic and draw work together.

Kinect Architecture.



(*NB This Image was taken from a website but I lost the reference. It was taken during the beta of the SDK NB*)

The Kinect sends the data collected to the NUI(Natural User Interface) which converts the raw data into useably data that the application then can use and interact with.

3.3 Implementation

The implementation of the project was done in two completely different ways from the prototype to the final submission. This was due to the fact that there was no official SDK and the

technology is so new. I will explain the Implementation of the prototype and the final submission. I will also provide code that was tried to implement features that did not work.

3.3.1 Prototype Implementation

During the early stages of development due to fact there was no Official SDK a beta version of the SDK was used. The code was written in C# using Microsoft Visual Studios 2010 and was built as a .WPF (Windows Presentation Foundation) application.

There are 4 main functions in the prototype. RGB Camera/ Camera Tilt/ Depth Perception/ Skeletal Tracking.

This code is in the appendix and is explained. It is commented to explain what each part dose.

3.3.2 Final Submission Implementation

The final submission of the project was built using the XNA framework and used the Official Kinect SDK. It was written in C# and uses the XNA Framework and the Kinect Official SDK.

Images tracking skeleton points

Here is a code segment taken from the Kinect.cs class.
All explanations of the code will be in the format that follows in these brackets
(//Comments/Explanation will be like this)
This code segment allows for mapping images to a point of the Kinect Skeleton. This code can be found in the avatar management region.

```
//The Head Image is held as a texture in the game
Texture2D HeadTexture;
Rectangle HeadRectangle;

//The position is held as an integer and a vector.

public int HeadX, HeadY;
public Vector2 HeadVector;


JointType HeadJoint = JointType.Head;
Creates HeadJoint variable which tracks to the head

//The follow function allows the head image to overlay the joint postion and keeps
it updated.
void updateHead()
        {
            if (activeSkeletonNumber == 0)
//This is saying if no skeleton is being tracked it places the image off the
screen so it is not visible.
            {
                HeadX = -200;
                HeadY = -200;
```

```
            }
            else
            {
                Joint jointHead = activeSkeleton.Joints[HeadJoint];
// if a skeleton is being tracked it places the image to the joint
                ColorImagePoint HeadPoint = theKinect.MapSkeletonPointToColor(
                    jointHead.Position,
                    ColorImageFormat.RgbResolution640x480Fps30);
//this is the method to get the postion of the video image of the point that is
tracked by the image
                HeadX = HeadPoint.X;
                HeadY = HeadPoint.Y;

            }

            HeadVector.X = HeadX;
            HeadVector.Y = HeadY;
//This is the update method that keeps the image tracking the point
            HeadRectangle.X = HeadX - HeadRectangle.Width / 2;
            HeadRectangle.Y = HeadY - HeadRectangle.Height / 2;
        }
```

3.3.3 Other Areas Explored

During the early stages of development an Open source version of the SDK was looked into this is call OpenNI (NI stands for Naturel Interface) Only tutorial's where completed using OpenNI. But it was not used as the Beta for the Microsoft Kinect SDK seemed to be the better option at the time.

3.3.4 Code tried that was not finished

A Menu System was attempted to be implement but unfortunately it could not be completed. The code I tried to implement is found in the appendix. The problem I had with the menu integration was getting the objects to be Visible or not Visible to tell if they are onscreen or not. This is done with a bools and gamestates I will provide the code segments for building the menu and working behind to tell when to change screens

3.4 Testing

Due to the fact that the Kinect is a natural interface all testing had to be done by trying out the game there was not specific tool used to test the system. The ways a list set out to check where certain aspects working

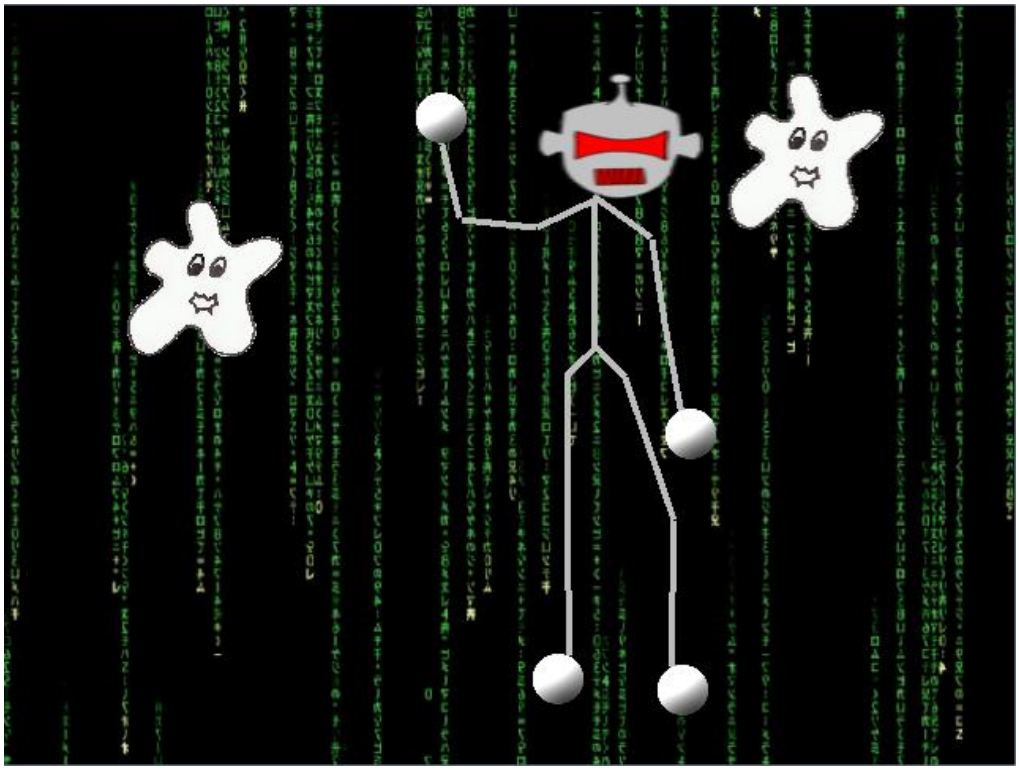
Object	Loading and moving across screen	Ok
Skeleton	Loading when person detected	Some problems with other objects being picked up
Collision Detection	Working with both hands	Some problems with unreachable points due to objects in the way
Sound Working	Yes sound on collision	OK

Avatar Loading	Yes	Images don't run flush
Error Detection	Yes	Display message when there is no Kinect detected
Points System	Not working	Not appearing onscreen
Menu	Not working	Not loading

This is an example of a check list I would have used during the different stages of development.

3.5 Graphical User Interface (GUI) Layout

sThere was no proper GUI added into the game and the most Kinect applications would have a NUI(Natural User Interface) Instead of a GUI as natural things like body movements or voice activation would act as inputs. This is a screenshot of the game as it was submitted with the detection of a person. It is simply the background and two of the enemy's shown loaded in and a players avatar in the middle game.



When the Player is detected the skeleton is drawn and Images are mapped to locations such as head Hands and feet. It is a natural interface as the hands are moved to remove the objects from screen.

3.6 Customer testing

A list of short question where done up to ask potential customers after trying the game would they have interest in this type of game or what changes or further development they would like to see be made.

Members of the public helped to test my game. Some members filled out a paper questionnaire and the same questionnaire was also done up on SurveyMonkey which other people filled out. The following is a link to the survey (http://www.surveymonkey.com/s/9GLGF6J).

This link was not publicly sent around it was sent in message to people who played the game.

Also a paper version can be found in the appendix and it is titled “Hack Attack Questionnaire”.

I have taken one example of a 22 year old to show what they thought of the game.

2. How often do you play video games?
Answer: Daily

3. How do you rate the playability of the game?
Answer: Good

4. How do you rate the Fun Factor of the game?
Answer: Good

5. How do you rate the graphics of the game?
Answer: Ok

6. How do you rate the overall game?
Answer: Good

7. What changes or improvements would you like to see. Or would you like to see it further developed?
Answer: i would like to see future developments, i think maybe a level system or some difficulty adjustment and maybe a soundtrack

8. Any Comments?
Answer: i like the game and i think judging by the laughs while playing its fun and cheerful which is what u want from a kinnect game

This other example is the answers of a 10 year old

2. How often do you play video games?
Answer: Daily

3. How do you rate the playability of the game?
Answer: Very Good

4. How do you rate the Fun Factor of the game?
Answer: Very Good

5. How do you rate the graphics of the game?
Answer: Good

6. How do you rate the overall game?
Answer: Very Good

7. What changes or improvements would you like to see. Or would you like to see it further developed?
Answer: yes I want to see more bad guys

8. Any Comments?
Answer: its super fun

From the results and my observations of people playing I feel the younger age group 13 and under have lots of fun with the game and love the jumping around aspect. Some of the older age group did not like the game as they felt embarrassed with people watching where other times a good laugh was made while people where playing and watching.

3.7 Evaluation

The system was evaluated through testing of my own. The checklist that was shown in the testing section above was to point out and check if there were any problems.

The response time of the skeleton where good and the collision detection worked well. Bit slow to load at times and picks up the skeleton well in the right conditions.

Missing a lot of component's to be considered a full game. But for what is working it works well.

My conclusion is even though I am not happy that it is not fully a finished game. I like the fact that I saw a lot of people get enjoyment out of it. That is the main purpose of a game if you ask me.

4 Conclusions

Advantages

Kinect is the start to a new form of gaming. It is also bringing about new wave of enhancements NUI (Natural user interface)which allows for many potential uses such as moving away from the standard input device e.g., controller, keyboard and mouse by allowing them to use hand movements and voice controls.

It has allowed me to learn about an upcoming technology during the early phase that I feel will have a big impact in the coming future. The gaming industry is a huge market and has many job opportunities. There are plenty of game development companies in Ireland. This experience has allowed me to see in what areas I need to increase my knowledge and the difficulties that arise during game development.

Disadvantages

At the Current stage there is still limited resource and during the development of this project nothing was certain. As I was trying to develop for Kinect, Microsoft where trying prefect and build the SDK. There is a slow increase of Kinect tutorials which means you are limited in what you can achieve. A new version of the SDK is due for release on the 15, June 2012. The SDK was changed twice during the beta phase and is going to be changed again within a few months of the initial official release. Another disadvantage was the fact that games are usually build by team of developers and wide skills set would have been required in order to produce a commercial quality game.

Opportunities

Due to the fact that the official version of the SDK allows for commercial, release with my current knowledge further game can be developed and released to potentially earn a profit.

There are also opportunities to further develop this game this will be discussed in the further development and research section.

There are many different areas in which the Kinect can be used. This means you are not limited to only designing games but also programmes which can be used in schools, hospitals and even the armed forces.

Limits

As you have seen there were limited resources available to me this and the fact that the code within the programme had to change with each new version of the SDK lead to the project having to be constantly restarted and left very little time to develop the game fully.

Even though the disadvantage and limitations of the project were great I am still happy that I chose to use the Kinect. This has been a valuable learning experience and I feel I have learned a great deal about time management, new technologies, mistakes I will hope not make again and that I enjoy this work and would like to pursue it in the future. From this project I can use the knowledge gained in my working life and maybe even developing something in my own time.

5 Further development or research

Unfortunately there was no menu or scoring system placed into the game this is another aspect that will be looked into. The menu system would include a selection for adjusting camera angles.

Voice recognition is an area that would be a great interest and would limit the need of any controller or keyboard inputs at all.

I would hope to put in more research and develop games that would be 3D so the Depth functionality could be put into play. Gesturer recognition is another area that could have great benefits.

Character selection that will allow the user to have some variety.

Multi-player functionality that will either be in the same room using one Kinect or over Xbox Live.

A game like this could be released as a part of a set of arcade games with more mini games added to increase its life span.

6 Bibliography

Webb, Jarrett and Ashley, James (2012). *Beginning Kinect Programming with the Microsoft Kinect SDK*. New York: Apress.

Kean, Sean and Hall, Jonathan and Perry, Phoenix (2012). *Meet The Kinect*. New York: Apress.

Miles, Rob (2012). *Using Kinect for Windows with XNA*. Hull: https://www.facultyresourcecenter.com/curriculum/pfv.aspx?ID=8938&c1=en-us&c2=0&Login=.

http://kinectxna.blogspot.com/ 9, May 2012

http://forums.create.msdn.com/forums/ 9, May 2012

http://msdn.microsoft.com/en-us/library/bb198548%28v=xnagamestudio.10%29.aspx 13, May 2012

http://xnagpa.net/index.php 7, May 2012

http://ross-warren.co.uk/pong-clone-in-xna-4-0-for-windows/ 2, May 2012

http://www.xnadevelopment.com/index.shtml 11, May 2012

http://rbwhitaker.wikidot.com/xna-tutorials 13, May 2012

http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx

http://channel9.msdn.com/Series/KinectQuickstart 14, April 2012

http://channel9.msdn.com/coding4fun/kinect 28, April 2012

7 Appendix

7.1 Project Proposal

Project Proposal

TITLE

Kinect Game (Title Undecided)

Sam Duff

x08794715

samduffirl@gmail.com

duff-man89@hotmail.com

BSc (Hons) in Software Systems

Date

23/Sep/11

Objectives

The main objective of this project is to provide a game that uses the core functionality of Microsoft Kinect's motion sensor.

The user will be able to enjoy the game by being the controller and will have a much greater interactive experience than just sitting there with a controller in hand. The user will be able to see their full body movements happen on screen. If necessary the Kinects voice commands may also be used in game.

Background

The reason I decided to create this application is because the Kinect motion senor is very new technology and has become a great interest to the hacking and developer community. It is promised to be the future of gaming. I have a keen interest in gaming which is another key factor to way I chose this as my project.

Kinect has a large market as it holds the Guinness world Record for "fastest selling consumer electronics device". It sold a massive 8 million units in the first 60 Days.

Technical Approach

Once I had my idea for the project, I began research on the technologies that will be used. The platform that my game will need to run on is an Xbox 360. Xbox is the platform that the Kinect was developed for. It is also possible to run it on computers with windows.

The development environment for the project is as follows

Hardware

- Kinect
- A Computer

Software

- Window 7
- Kinect SDK
- Visual studio 2010 Express
- Microsoft .NET framework 4.0
- Programming language used C++/C#

Special resources required

I will require sources to help me with development of the game there are plenty of online sources that will help me. Here are examples of some online sources that can be used.

http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/guides.aspx

http://www.kinecteducation.com/forumdisplay.php?12-Kinect-Programming-Resources

http://www.developerfusion.com/t/kinect/

Here is a list of books that can be used

Hacking with the Microsoft Kinect Beta SDK: A guided tour through the world of NUI [Paperback]

David Bates (Author)

Meet the Kinect: Programming and Scripting Natural User Interfaces [Paperback]

Sean Kean (Author), Jonathan Hall (Author), Phoenix Perry (Author)

Beginning Kinect Programming with the Microsoft Kinect SDK [Paperback]

Jarrett Webb (Author), James Ashley (Author)

Making Things See: 3D vision with Kinect, Processing, and Arduino [Paperback]

Greg Borenstein (Author)

Project Plan

Project Milestones	**Completion Date**
Project Proposal	23rd Sep
Requirement Specification	7th Oct
Completing Tutorials to learn how to develop using Kinect.	21st Oct
Draw up the design of the game	4th Nov
Develop Game	14th Dec
Prototype	15th Dec
Continuing Development towards Beta	29th Apr
Beta	30th Apr
Continuing Development towards Finished Project	4th May

Bug Fixing	17th May
Documentation	17th May
Project Submission	18th May

Technical Details

The programming language used will be C#

http://openkinect.org/wiki/Main_Page

The link above provides libraries that can be used during development of my game.

Evaluation

My plan for testing the application is to have friends and family test the game for me under normal conditions and get them to provide feedback on problems they had with the game.

Consultation 1

Keith Maycock

During my meeting with Keith he said with the idea I have I should look at developing an avatar that works around your skeleton and to keep away from features like finger tracking and to stick skeleton tracking.

Consultation 2

Michael Bradford

During my meeting with Michael he said to keep the idea simple as it will be a hard project to complete. He said a game that is more augmented reality might be a good idea.

So it will be you playing and your current surroundings with the game added into the screen.

Proposed Supervisor

Names of academic staff member that has agreed to act as a supervisor for this project.

Sam Duff 27/09/2011

Signature of student and date

7.2 Requirement Specification

Title Requirements Specification (RS)

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
14/10/2005	1	Create	AB	X	X
21/10/05	2	Update	CD		

Distribution List

Name	Title	Version
Keith Maycock	Lecturer II	2

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

Table of Contents

1 Introduction 34

1.1 Purpose 34

1.2 Project Scope 34

1.3 Definitions, Acronyms, and Abbreviations 34

2 User requirements definition 34

3 System architecture 34

4 Requirements specification 35

4.1 Physical environment requirements 35

4.1.1 Requirement 1 <Play Space> 35

4.1.2 Description & Priority 35

4.1.3 Requirement 2 <Lighting> 35

4.1.4 Description & Priority 35

4.2 Interface requirements 35

4.3 Functional requirements 35

4.3.1 Requirement 1 <Start Game> 35

4.3.1.1 Description & Priority 35

4.3.1.2 Requirement Activation 35

4.3.2 Requirement 2 <View High Score> 36

4.3.2.1 Description & Priority 36

4.3.2.2 Requirement Activation 36

4.3.3 Requirement 3 <Change Avatar> 36

4.3.3.1 Description & Priority 36

4.3.3.2 Requirement Activation 36

4.4 Documentation requirements 36

4.4.1 Requirement 1 <User Manual> 36

4.4.2 Description & Priority 36

4.5 Non-Functional Requirements 36

4.5.1 Performance/Response time requirement 36

5 System evolution 36

6 Appendices 37

6.1 Use case diagram 37
6.1 Use case 1 37
6.2 Use case 2 38
6.3 Use case 3 40
6.4 Figure A 42

Introduction

Purpose

The purpose of this document is to set out the requirements for the development of a game that uses Microsoft's Kinect motion sensor

The intended customers are people who have a love for games and are more interested in a more interactive style of playing as the players body will be the controller. The age bracket is more for a younger audience

Project Scope

The scope of the project is to develop a game that uses the functionality of the Kinect motion sensor. The game shall have a fully interactive interface using hand or arm motion. It will also have a score system to allow the players to try and beat there score.

Definitions, Acronyms, and Abbreviations

SDK Software Development Kit

User requirements definition

The objective for the game is as follows

- A Menu for the user to make selections.
- The game will have an avatar on screen that will copy all of the user's actions.
- Multiple avatars to chose from
- The avatar will interact with features of the game to allow it to gain a score, see how long it can keep a task going for or progress to the next level.
- A leader board to save individual high scores

Wish List

- The game will have a voice control to make selections in menu.
- Import avatars by scanning objects using Kinect

System architecture

The System architecture consists of the following components: User Interface, Xbox 360, Kinect and Database for keeping high scores. The main architecture of the system will follow the MVC pattern to ensure that the components of the system are designed independently. Figure A in the appendices shows a high level protocol for the project. The arrows in the diagram shows the flow of communication between the user and devices.The game will be written using the language C#.

Requirements specification

The game is being designed to run on well-established components.

How the Kinect works: The device features an "RGB camera, depth sensor and multi-array microphone running proprietary software", which provide full-body 3D motion capture, facial recognition and voice recognition capabilities

Physical environment requirements

Requirement 1 <Play Space>

At least 6 feet between the user and the Kinect sensor is needed.

Description & Priority

To pick up your full body there is a requirement of 6 feet between the player and the Kinect sensor. This is important as the Kinect will not be able to see your full body and will not be able to recreate body movements.

Requirement 2 <Lighting>

An adequate amount of light is required.

Description & Priority

If proper lighting is not available the sensor may not be able to track body movements

Interface requirements

A GUI will have to be set up for the menu and game interface.

All input will be done using a Keyboard/Mouse(PC) Controller(Xbox) but mainly through the Kinect by body movements.

Functional requirements

The following specify the functional requirements and what they should accomplish.

Requirement 1 <Start Game>

Description & Priority

The user starts to play the game.

Requirement Activation

When Start Game is selected from the main menu

Requirement 2 <View High Score>

Description & Priority

The user can view if he got a high score

Requirement Activation

When Leader boards is selected from the main menu

Requirement 3 <Change Avatar>

Description & Priority

This will allow the user to change which avatar represents them in the game

Requirement Activation

When characters is selected from the main menu

Documentation requirements

A brief user manual will be created to show what poses cause action like pausing the game but instructions will also happen in the game.

Requirement 1 <User Manual>

Description & Priority

The user manual will have a list of instructions and poses that preform different actions.

Non-Functional Requirements

Performance/Response time requirement

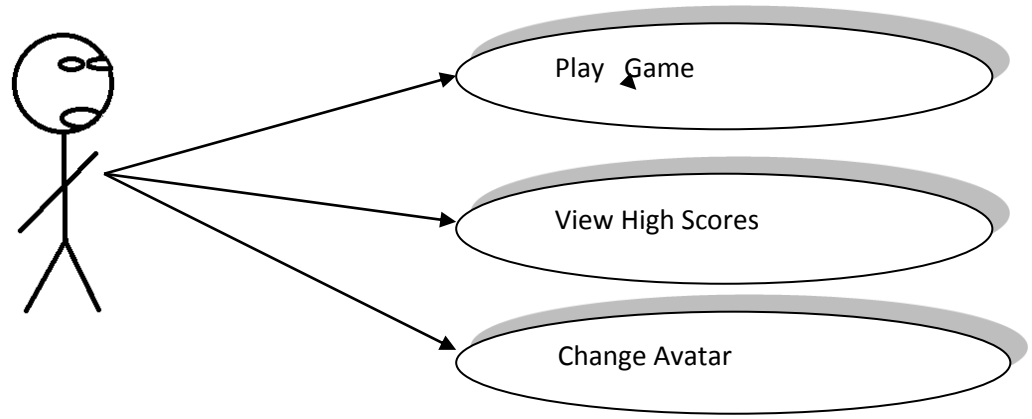
Response time is an important factor as the game will involve the user to react to moving objects on screen. If there is a delay between the user movements and game movements the user will probably be frustrated and will not enjoy the game.

System evolution

Over time features such as voice control, more minigames and live multiplayer can be added. Such features of the Kinect may evolve. New features such as finger tracking are coming into use so the game could evolve to use features like this.

Appendices

Use case diagram



Use case 1

Use case

Play Game

Scope

How the user starts playing the game.

Description

This use case describes the process of how the player plays the game and terminates the system

Flow Description

Precondition

The system is in initialisation mode……..

Activation

This use case starts when an the user appears in front of the Kinect sensor and waves to allow the system know they are ready to play

Main flow

1. The system identifies that there is a person standing in front of the sensor<E1>
2. The system puts a default avatar is put in place.
3. The <Actor> makes a menu selection by moving there hand to the point of that start game<A1><A2>
4. The system starts the game

5. The <Actor> plays the game.
6. The system goes back to the menu when the game is lost.

Alternate flow

A1 :

1. The actor picks Change avatar
2. The system bring to avatar page
3. The actor picks an avatar
4. The system goes back to main menu
5. Use case continues at point 3

A2 :

1. The actor picks View high Scores
2. The system bring to High Scores page
3. The actor picks to go back to the main menu
4. The system goes back to main menu
5. Use case continues at point 3

Exceptional flow

E1 :

1. The system cannot identify there is a person and dose not continue

Termination

The system terminates when no player is visible to sensor.

Post condition

The system goes into a wait state

Use case 2

Use case

View high Scores

Scope

How the user Views the high scores

Description

This use case describes the process of how the views the high scores and terminates the system

Flow Description

Precondition

The system is in initialisation mode……..

Activation

This use case starts when an the user appears in front of the Kinect sensor and waves to allow the system know they are ready to play

Main flow

1. The system identifies that there is a person standing in front of the sensor<E1>
2. The system puts a default avatar is put in place.
3. The <Actor> makes a menu selection by moving their hand to the point of that View high Scores <A1><A2>
4. The system bring to High Scores page
5. The actor picks to go back to the main menu
6. The system goes back to main menu

Alternate flow

A1 :

6. The actor picks Change avatar
7. The system bring to avatar page
8. The actor picks an avatar
9. The system goes back to main menu
10. Use case continues at point 3

A2 :

6. The actor picks Play Game
7. The system starts the game
8. The <Actor> plays the game.
9. The system goes back to the menu when the game is lost
10. Use case continues at point 3

Exceptional flow

E1 :

1.The system cannot identify there is a person and dose not continue

Termination

The system terminates when no player is visible to sensor.

Post condition

The system goes into a wait state

Use case 3

Use case

View high Scores

Scope

How the user Views the high scores

Description

This use case describes the process of how the views the high scores and terminates the system

Flow Description

Precondition

The system is in initialisation mode……..

Activation

This use case starts when an the user appears in front of the Kinect sensor and waves to allow the system know they are ready to play

Main flow

7. The system identifies that there is a person standing in front of the sensor<E1>
8. The system puts a default avatar is put in place.
9. The <Actor> makes a menu selection by moving their hand to the point of that Change Avatar <A1><A2>
10. The system bring to Change Avater page
11. The actor picks an Avatar
12. The system goes back to main menu

Alternate flow

A1 :

1. The actor picks View high Scores
2. The system bring to High Scores page
3. The actor picks to go back to the main menu
4. The system goes back to main menu
5. Use case continues at point 4

A2 :

6. The actor picks Play Game
7. The system starts the game
8. The <Actor> plays the game.
9. The system goes back to the menu when the game is lost
10. Use case continues at point 3

Exceptional flow

E1 :

1.The system cannot identify there is a person and dose not continue

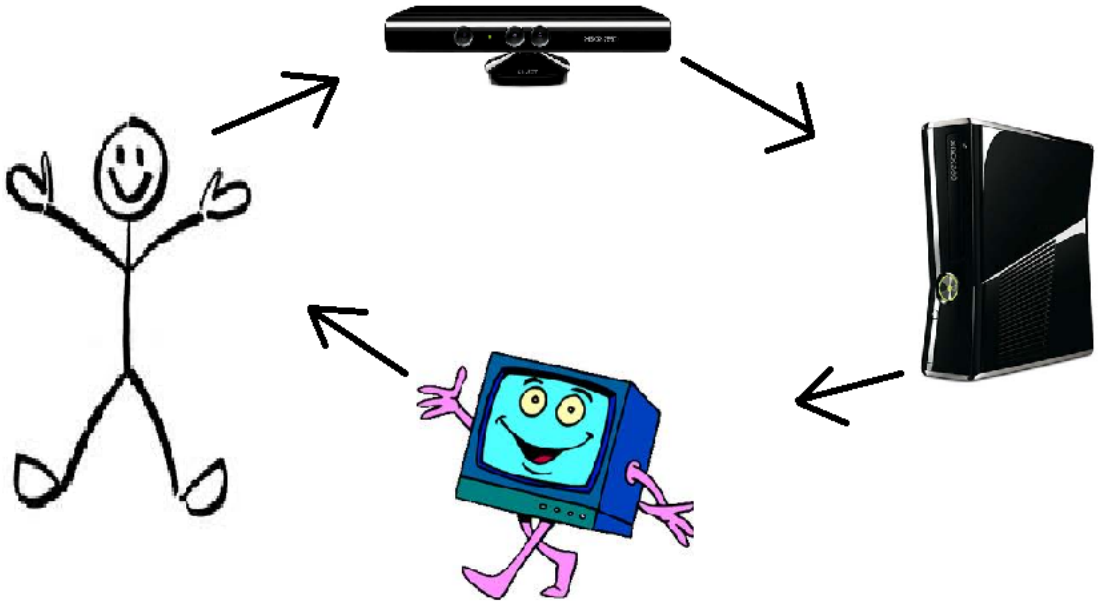
Termination

The system terminates when no player is visible to sensor.

Post condition

The system goes into a wait state

Figure A



7.3 Monthly log book

Software Project

Reflective Journals

Student Name: Sam Duff

Student Number: x08794715

Reflective Journal 1

So far in my time spent working on this project I have

Set up SDK's

As there where many ways to go about working on the project I downloaded the official Beta Kinect SDK.

Software needed with the beta SDK to build the project was downloaded

Microsoft Visual Studio

Microsoft .NET Framework 4

There is also an open source version OpenNI which was also downloaded and explored and researched.

Tutorials

Tutorials and research was done to prepare for development.

Reflective Journal 2

Since the last journal I have spent my time

Kinect Research

I chose to do my development with the beta version of the SDK and the follow was done.

Learned the basics of

- skeletal tracking
- reading camera data from the Kinect sensor
- depth data from Kinect
- reading audio data from the Kinect microphone array

Also spent time reviewing code samples that where found.

Problems

- I had problems with the compatibility of version of software being used which had to be resolved.
- Books are currently not available on development with Kinect and the earliest release date of one is Dec/2011

Reflective Journal 3

Since the last journal I have spent my time

Prototype

- Started writing code for my project. Implemented what I have learned to start working with Kinect using the beta version of the SDK.
- I have only implemented features that may be relevant. I did implement depth data in case the Game is to be turned 3D. Did not include any voice features.

Presentation

- Attempted to make a program that would allow gestures to register as keyboard keys. To allow me to go through the presentation without touching the keyboard. I was unsuccessful with this.

Problems

- Little online resources.
- Books are currently not available on development with Kinect and the earliest release date of one is Dec/2011

Reflective Journal 4

Since the last journal I have spent my time

Research

- Meet the Kinect: An Introduction to Programming Natural User Interfaces (Technology in Action) by Sean Kean, Jonathan Hall and Phoenix Perry. Is the first book released about development with Kinect. Went through some of the book. Still reading it. Dose not look as it could be much help.
- Preformed research as the official SDK is due to be released on the 1st of Feb and there will be changes from the current Beta SDK which I am currently using.

Problems

- Due to exams not much work could be completed Since the last Journal

Reflective Journal 5

Since the last journal I have spent my time

Research

Went through the book Meet the Kinect to expand knowledge of the technology. It was not very helpful in terms of what knowledge I need to acquire.

Updated from the beta version of the SDK to the official version of the SDK.

Reviewed changes made to SDK

Went through tutorial's using the new official SDK to see the code changes.

Reflective Journal 6

Since the last journal I have spent my time

Found tutorial's on how to write use XNA with Kinect.

Went through the tutorials on how to work Kinect with XNA

Started reading. I purchased Beginning Kinect programming with the Microsoft Kinect SDK by Jerrett Webb and James Ashley. The only other Kinect book available at the moment.

Reflective Journal 7

Since the last journal I have spent my time

Developed what I have learned from the tutorials I found into my final project. This project was using XNA. Further research had to be spent learning how XNA works and what ways I could use it. And practicing tutorials using just XNA without the Kinect

7.4 Other material Used

Any other reference material used in the project for example evaluation surveys etc.

CD containing code should be glued to the technical report.

7.5 Prototype Code

7.5.1 MainWindow.xaml.cs

These are the using statements that call the different frameworks used

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Research.Kinect.Audio; //Added in to load the Kinect Audio Functions

using Microsoft.Research.Kinect.Nui; // Added in to load the Kinect Natural User Interface
functions

using Coding4Fun.Kinect.Wpf; // The Coding for fun toolkit adds in adds in shorted methods

namespace KinectProject
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        Runtime nui = new Runtime();//Kinect Runtime

        //Sets each of the 20 joints tracked to a different colors
        Dictionary<JointID, Brush> jointColors = new Dictionary<JointID, Brush>() {
            {JointID.HipCenter, new SolidColorBrush(Color.FromRgb(169, 176, 155))},
            {JointID.Spine, new SolidColorBrush(Color.FromRgb(169, 176, 155))},
            {JointID.ShoulderCenter, new SolidColorBrush(Color.FromRgb(168, 230,
29))},
            {JointID.Head, new SolidColorBrush(Color.FromRgb(200, 0,   0))},
            {JointID.ShoulderLeft, new SolidColorBrush(Color.FromRgb(79,  84,  33))},
            {JointID.ElbowLeft, new SolidColorBrush(Color.FromRgb(84,  33,  42))},
            {JointID.WristLeft, new SolidColorBrush(Color.FromRgb(255, 126, 0))},
            {JointID.HandLeft, new SolidColorBrush(Color.FromRgb(215,  86, 0))},
            {JointID.ShoulderRight, new SolidColorBrush(Color.FromRgb(33,  79,  84))},
            {JointID.ElbowRight, new SolidColorBrush(Color.FromRgb(33,  33,  84))},
            {JointID.WristRight, new SolidColorBrush(Color.FromRgb(77,  109, 243))},
            {JointID.HandRight, new SolidColorBrush(Color.FromRgb(37,   69, 243))},
            {JointID.HipLeft, new SolidColorBrush(Color.FromRgb(77,  109, 243))},
            {JointID.KneeLeft, new SolidColorBrush(Color.FromRgb(69,  33,  84))},
```

```
            {JointID.AnkleLeft, new SolidColorBrush(Color.FromRgb(229, 170, 122))},
            {JointID.FootLeft, new SolidColorBrush(Color.FromRgb(255, 126, 0))},
            {JointID.HipRight, new SolidColorBrush(Color.FromRgb(181, 165, 213))},
            {JointID.KneeRight, new SolidColorBrush(Color.FromRgb(71, 222,  76))},
            {JointID.AnkleRight, new SolidColorBrush(Color.FromRgb(245, 228, 156))},
            {JointID.FootRight, new SolidColorBrush(Color.FromRgb(77,  109, 243))}
        };

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
//Starts up the runtime for Color = StandedRgb/ Depth= Depth Sensor / DepthAndPlayerIndex  = depth data of multipule skeletons/SkeletalTracking  = Tracks Skeletons picks up all the 20 joints
            nui.Initialize(RuntimeOptions.UseColor | RuntimeOptions.UseDepth | RuntimeOptions.UseDepthAndPlayerIndex | RuntimeOptions.UseSkeletalTracking);//Runtime object with Color, Depth and SkeletalTracking information

//Create event handlers which handles video/depth/skeleton
            nui.VideoFrameReady += new EventHandler<ImageFrameReadyEventArgs>(nui_VideoFrameReady);//VidoeFrame Ready event for video
            nui.DepthFrameReady += new EventHandler<ImageFrameReadyEventArgs>(nui_DepthFrameReady);//VidoeFrame Ready event for depth
            //add event to receive skeleton data
            nui.SkeletonFrameReady += new EventHandler<SkeletonFrameReadyEventArgs>(nui_SkeletonFrameReady);
            //Runs starts open events
            nui.VideoStream.Open(ImageStreamType.Video, 2, ImageResolution.Resolution640x480, ImageType.Color);//Starts(opens) event video
            nui.DepthStream.Open(ImageStreamType.Depth, 2, ImageResolution.Resolution640x480, ImageType.Depth);//Starts(opens) event depth

        }
//This is the code that takes the skeleton data and coverts it to the play space available
        private Point getDisplayPosition(Joint joint)
        {
            float depthX, depthY;
            nui.SkeletonEngine.SkeletonToDepthImage(joint.Position, out depthX, out depthY);
            depthX = depthX * 320; //convert to 320, 240 space
            depthY = depthY * 240; //convert to 320, 240 space
            int colorX, colorY;
            ImageViewArea iv = new ImageViewArea();
            // only ImageResolution.Resolution640x480 is supported at this point

nui.NuiCamera.GetColorPixelCoordinatesFromDepthPixel(ImageResolution.Resolution640x480, iv, (int)depthX, (int)depthY, (short)0, out colorX, out colorY);

            // map back to skeleton.Width & skeleton.Height
            return new Point((int)(skeleton.Width * colorX / 640.0), (int)(skeleton.Height * colorY / 480));
        }

        Polyline getBodySegment(Microsoft.Research.Kinect.Nui.JointsCollection joints, Brush brush, params JointID[] ids)
        {
            PointCollection points = new PointCollection(ids.Length);
            for (int i = 0; i < ids.Length; ++i)
            {
```

```
                points.Add(getDisplayPosition(joints[ids[i]]));
            }

            Polyline polyline = new Polyline();
            polyline.Points = points;
            polyline.Stroke = brush;
            polyline.StrokeThickness = 1;
            return polyline;
        }

        void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
//if there is multiple skeletons changes the color of the different skeletons
            SkeletonFrame skeletonFrame = e.SkeletonFrame;
            int iSkeleton = 0;
            Brush[] brushes = new Brush[6];
            brushes[0] = new SolidColorBrush(Color.FromRgb(255, 0, 0));
            brushes[1] = new SolidColorBrush(Color.FromRgb(0, 255, 0));
            brushes[2] = new SolidColorBrush(Color.FromRgb(64, 255, 255));
            brushes[3] = new SolidColorBrush(Color.FromRgb(255, 255, 64));
            brushes[4] = new SolidColorBrush(Color.FromRgb(255, 64, 255));
            brushes[5] = new SolidColorBrush(Color.FromRgb(128, 128, 255));

            skeleton.Children.Clear();
            foreach (SkeletonData data in skeletonFrame.Skeletons)
            {
                if (SkeletonTrackingState.Tracked == data.TrackingState)
                {
                    // Draw bones between Joints
                    Brush brush = brushes[iSkeleton % brushes.Length];
                    skeleton.Children.Add(getBodySegment(data.Joints, brush,
JointID.HipCenter, JointID.Spine, JointID.ShoulderCenter, JointID.Head));
                    skeleton.Children.Add(getBodySegment(data.Joints, brush,
JointID.ShoulderCenter, JointID.ShoulderLeft, JointID.ElbowLeft, JointID.WristLeft,
JointID.HandLeft));
                    skeleton.Children.Add(getBodySegment(data.Joints, brush,
JointID.ShoulderCenter, JointID.ShoulderRight, JointID.ElbowRight, JointID.WristRight,
JointID.HandRight));
                    skeleton.Children.Add(getBodySegment(data.Joints, brush,
JointID.HipCenter, JointID.HipLeft, JointID.KneeLeft, JointID.AnkleLeft,
JointID.FootLeft));
                    skeleton.Children.Add(getBodySegment(data.Joints, brush,
JointID.HipCenter, JointID.HipRight, JointID.KneeRight, JointID.AnkleRight,
JointID.FootRight));

                    // Draw joints
                    foreach (Joint joint in data.Joints)
                    {
                        Point jointPos = getDisplayPosition(joint);
                        Line jointLine = new Line();
                        jointLine.X1 = jointPos.X - 3;
                        jointLine.X2 = jointLine.X1 + 6;
                        jointLine.Y1 = jointLine.Y2 = jointPos.Y;
                        jointLine.Stroke = jointColors[joint.ID];
                        jointLine.StrokeThickness = 6;
                        skeleton.Children.Add(jointLine);
                    }
                }

                iSkeleton++;
            } // for each skeleton
```

```
        }
        private void button1_Click(object sender, RoutedEventArgs e)
        {
            //Set angle to the slider value
            nui.NuiCamera.ElevationAngle = (int)slider.Value;
        }

        void nui_DepthFrameReady(object sender, ImageFrameReadyEventArgs e)
        {
            //throw new NotImplementedException();
            //Use Coding4Fun class on ImageFrame class for Depth
            dep.Source = e.ImageFrame.ToBitmapSource();
        }

        void nui_VideoFrameReady(object sender, ImageFrameReadyEventArgs e)
        {

            // Creates BitmapSource for Video
            PlanarImage image = e.ImageFrame.Image;
            vid.Source = BitmapSource.Create(image.Width, image.Height, 96, 96,
PixelFormats.Bgr32, null, image.Bits, image.Width * image.BytesPerPixel);
        }


        private void Window_Closed(object sender, EventArgs e)
        {
            //stop Events clears everthing
            nui.Uninitialize();
        }
    }
}
```

7.5.2 MainWindow.xaml

```
//The below code is what sets the layout for the GUI shown below
<Window x:Class="KinectProject.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="750" Width="1000" Loaded="Window_Loaded"
Closed="Window_Closed">
    <Grid>
        <Image Height="150" HorizontalAlignment="Left" Name="vid" Stretch="Fill"
VerticalAlignment="Top" Width="198" Margin="39,21,0,0" /> //Sets the frame for the
video to 150X198 and adds a margin. You can see this is the box on the top left in the
GUI layout

        <Slider Height="150" HorizontalAlignment="Left" Margin="445,21,0,0"
Name="slider" VerticalAlignment="Top" Width="50" Orientation="Vertical" Maximum="27"
Minimum="-27" SmallChange="1" TickPlacement="TopLeft" Interval="1"
IsSnapToTickEnabled="True" /> //Adds in the slider that allows the degree of the
Camera to be set

        <Label Content="{Binding Path=Value, ElementName=slider}" Height="61"
HorizontalAlignment="Right" Margin="0,67,416,0" Name="label1" VerticalAlignment="Top"
FontSize="36" Width="60" /> //Adds the Label to show what dregee the camera angle is
set at
```

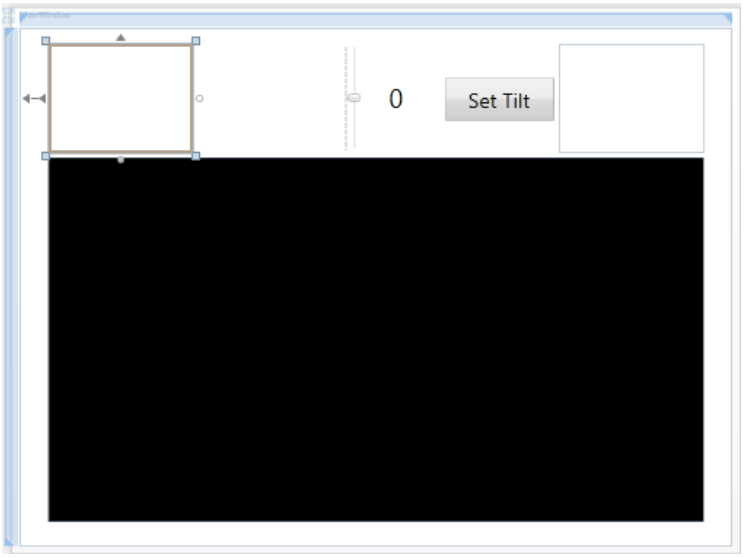
```
        <Button Content="Set Tilt" Height="61" HorizontalAlignment="Left"
Margin="583,67,0,0" Name="button1" VerticalAlignment="Top" Width="150" FontSize="28"
Click="button1_Click" /> //Adds the Button that actives the senor to move to the angle
Set

        <Image Height="150" HorizontalAlignment="Left" Margin="739,21,0,0" Name="dep"
Stretch="Fill" VerticalAlignment="Top" Width="200" /> //Adds Image to show depth
information

        <Canvas Width="900" Height="500" Name="skeleton" Margin="39,178,39,84"
Background="Black" ClipToBounds="True"></Canvas> //Adds the main area that the
Skeleton tracking shows up in.

    </Grid>
</Window>
```

7.5.3 GUI Layout for Prototype



7.6 Attempted Code

7.6.1 Input.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework.Input;

namespace XNAGameKinect
{
    public class Input
    {
        private KeyboardState keyboardState;
        private KeyboardState lastState;

        public Input()
        {
            keyboardState = Keyboard.GetState();
            lastState = keyboardState;
        }

        public void Update()
        {
            lastState = keyboardState;
            keyboardState = Keyboard.GetState();
        }

        public bool RightUp
        {
            get
            {
                if (Kinect.gamestate == Kinect.GameStates.Menu)
                {
                    return keyboardState.IsKeyDown(Keys.Up) &&
lastState.IsKeyUp(Keys.Up);
                }
                else
                {
                    return keyboardState.IsKeyDown(Keys.Up);
                }
            }
        }

        public bool RightDown
        {
            get
            {
                if (Kinect.gamestate == Kinect.GameStates.Menu)
                {
                    return keyboardState.IsKeyDown(Keys.Down) &&
lastState.IsKeyUp(Keys.Down);
                }
                else
                {
                    return keyboardState.IsKeyDown(Keys.Down);
                }
            }
        }
```

```
        public bool LeftUp
        {
            get
            {
                if (Kinect.gamestate == Kinect.GameStates.Menu)
                {
                    return keyboardState.IsKeyDown(Keys.W) &&
lastState.IsKeyUp(Keys.W);
                }
                else
                {
                    return keyboardState.IsKeyDown(Keys.W);
                }
            }
        }

        public bool LeftDown
        {
            get
            {
                if (Kinect.gamestate == Kinect.GameStates.Menu)
                {
                    return keyboardState.IsKeyDown(Keys.S) &&
lastState.IsKeyUp(Keys.S);
                }
                else
                {
                    return keyboardState.IsKeyDown(Keys.S);
                }
            }
        }

        public bool MenuSelect
        {
            get
            {
                return keyboardState.IsKeyDown(Keys.Enter) &&
lastState.IsKeyUp(Keys.Enter);
            }
        }
    }
}
```

This Code was to allow me to have Keyboard States when using the Menu

7.6.2 Menu.cs

```
namespace XNAGameKinect
{
    using System.Collections.Generic;
    using Microsoft.Xna.Framework;
    using Microsoft.Xna.Framework.Graphics;

    public class Menu
    {

        private List<string> MenuItems;
        private int iterator;
        public string InfoText { get; set; }
        public string Title { get; set; }
```

```
        public int Iterator
        {
            get
            {
                return iterator;
            }
            set
            {
                iterator = value;
                if (iterator > MenuItems.Count - 1) iterator = MenuItems.Count - 1;
                if (iterator < 0) iterator = 0;
            }
        }

        public Menu()
        {
            Title = "Hack Attack";
            MenuItems = new List<string>();
            MenuItems.Add("Play Game");
            MenuItems.Add("Exit Game");
            Iterator = 0;
            InfoText = string.Empty;
        }

        public int GetNumberOfOptions()
        {
            return MenuItems.Count;
        }

        public string GetItem(int index)
        {
            return MenuItems[index];
        }

        public void DrawMenu(SpriteBatch batch, int screenWidth, SpriteFont arial)
        {
            batch.DrawString(arial, Title, new Vector2(screenWidth / 2 -
arial.MeasureString(Title).X / 2, 20), Color.White);
            int yPos = 100;
            for (int i = 0; i < GetNumberOfOptions(); i++)
            {
                Color colour = Color.White;
                if (i == Iterator)
                {
                    colour = Color.Gray;
                }
                batch.DrawString(arial, GetItem(i), new Vector2(screenWidth / 2 -
arial.MeasureString(GetItem(i)).X / 2, yPos), colour);
                yPos += 50;
            }
        }

        public void DrawEndScreen(SpriteBatch batch, int screenWidth, SpriteFont
arial)
        {
            batch.DrawString(arial, InfoText, new Vector2(screenWidth / 2 -
arial.MeasureString(InfoText).X / 2, 300), Color.White);
            string prompt = "Press Enter to Continue";
            batch.DrawString(arial, prompt, new Vector2(screenWidth / 2 -
arial.MeasureString(prompt).X / 2, 400), Color.White);
        }
```

```
    }
}
```
The Menu.cs builds a menu that highlights selections.

Hack Attack Questionnaire

What age are you?

How often would you play games? Please tick box below.

Daily		Sometimes	
Weekly		Never	

On a Scale of 1 – 5, 1 been the lowest rating and 5 been the highest rating, how would you rate the following after playing Hack Attack.

Playability

Fun Factor

Graphics

Overall Game

What changes or improvement would you like to see with this game and would you like to see this game further developed?

Further Comments

7.7 Drawn Images that where scanned and edited with gimp